Integrarea sistemelor informatice

Suport curs nr. 4

Programator >> Arhitect

Integrarea aplicațiilor software

2024-2025

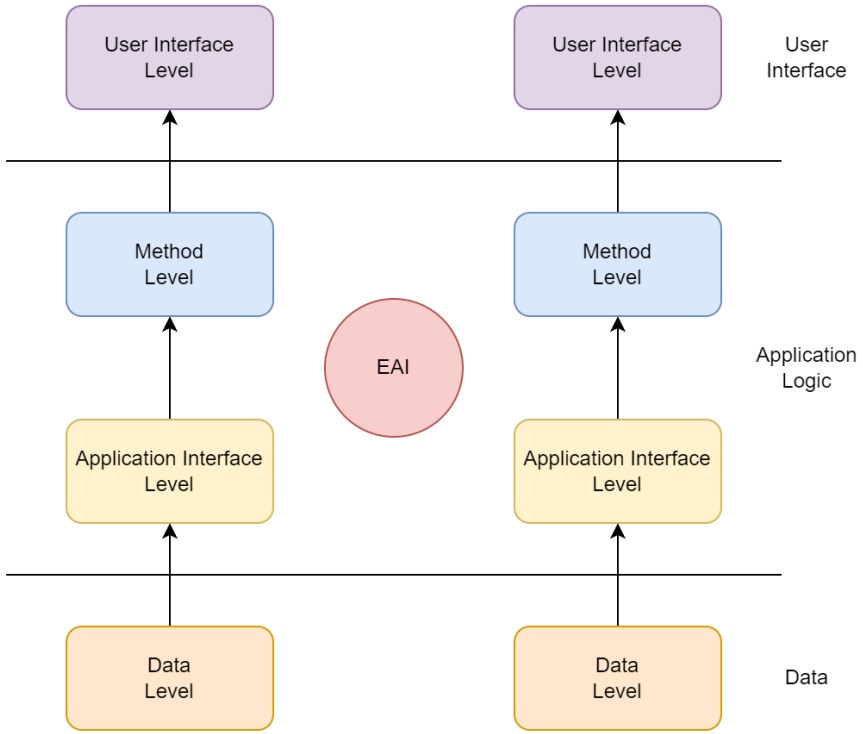
C4 – Integrarea aplicațiilor software

Obiective

- Identificarea modelelor arhitecturale la nivel de aplicație
- Înțelegerea conceptului de framework
- Proiectarea interfețelor API pentru integrarea aplicațiilor software
- Analiza sistemelor de aplicații integrate – studiu de caz

Recapitulare – niveluri de integrare

- **EAI** – Enterprise Application Integration
- Niveluri de integrare
  - Nivelul **de date**
  - Nivelul **funcțional** / logică aplicație
  - Nivelul **interfeței utilizator**

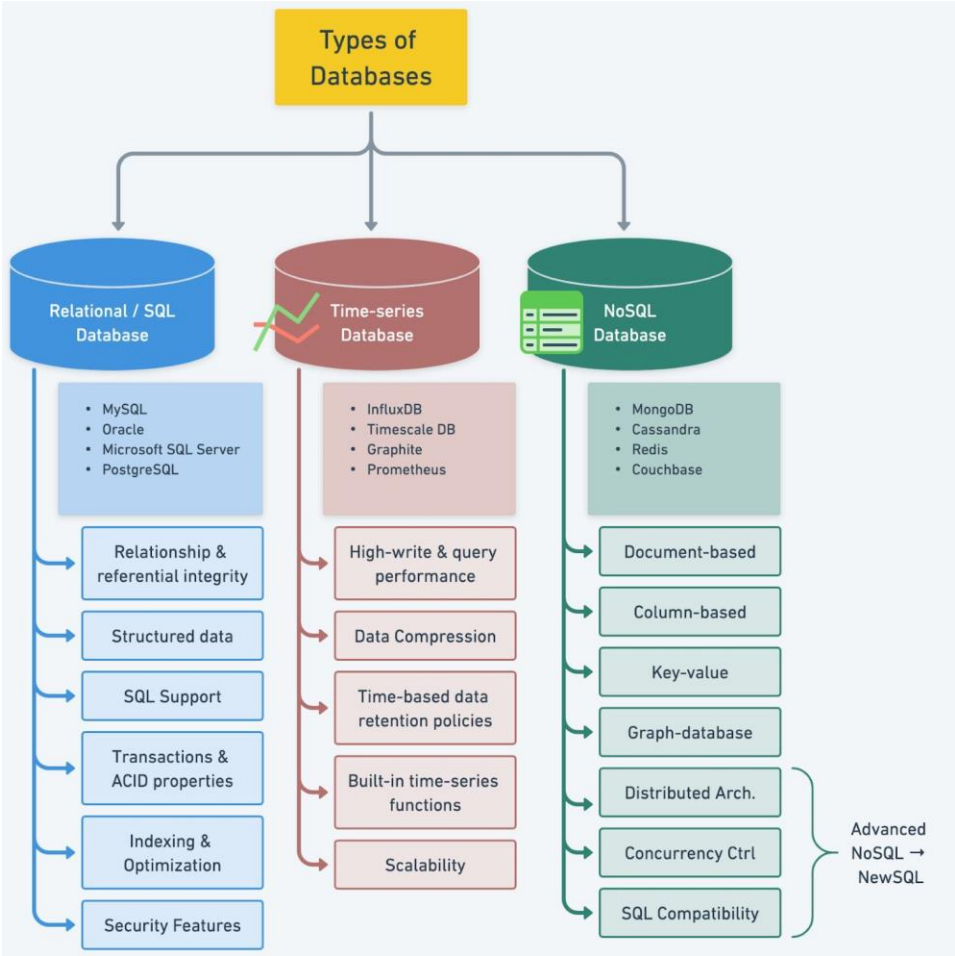


Sursa: Enterprise Application Integration (EAI), CIO Wiki

Nivelul de date

Baze de date

- Relaționale / SQL
- Nerelaționale / NoSQL
  - Documente
  - Grafuri
  - Cheie-valoare (en. key-value)
- Serii de timp / Time-series

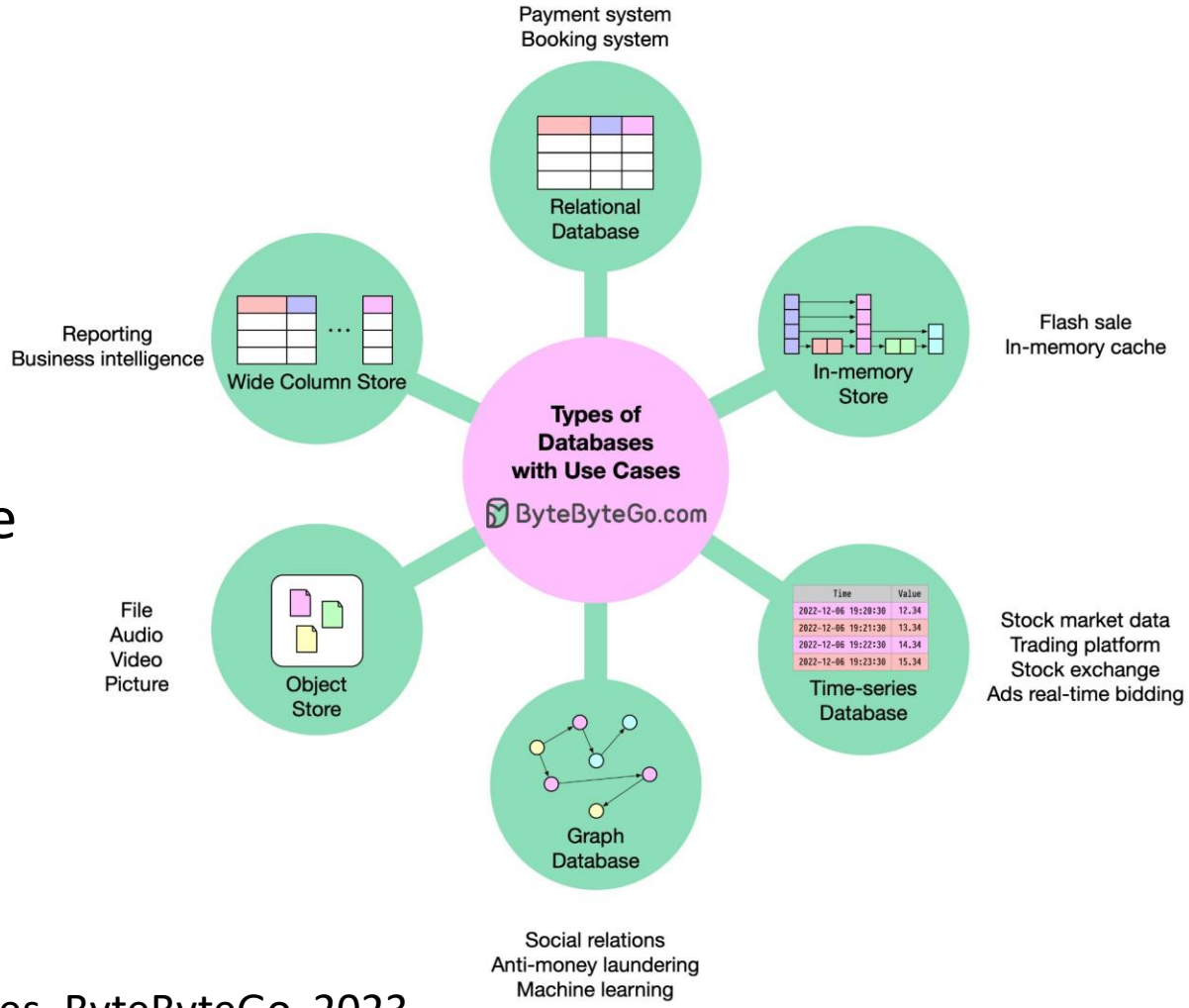


Sursa: ByteByteGo, 2023

Nivelul de date

Cum alegem tipul **bazei de date** în funcție de tipul aplicației

- Relaționale / SQL – aplicații uzuale
- In-memory – operații rapide
- Time-series – operații cu serii de timp
- Grafuri – relații complexe între obiecte nestructurate
- Documente – date imutabile de dimensiuni mari
- Coloane largi – big data, analytics, raportări



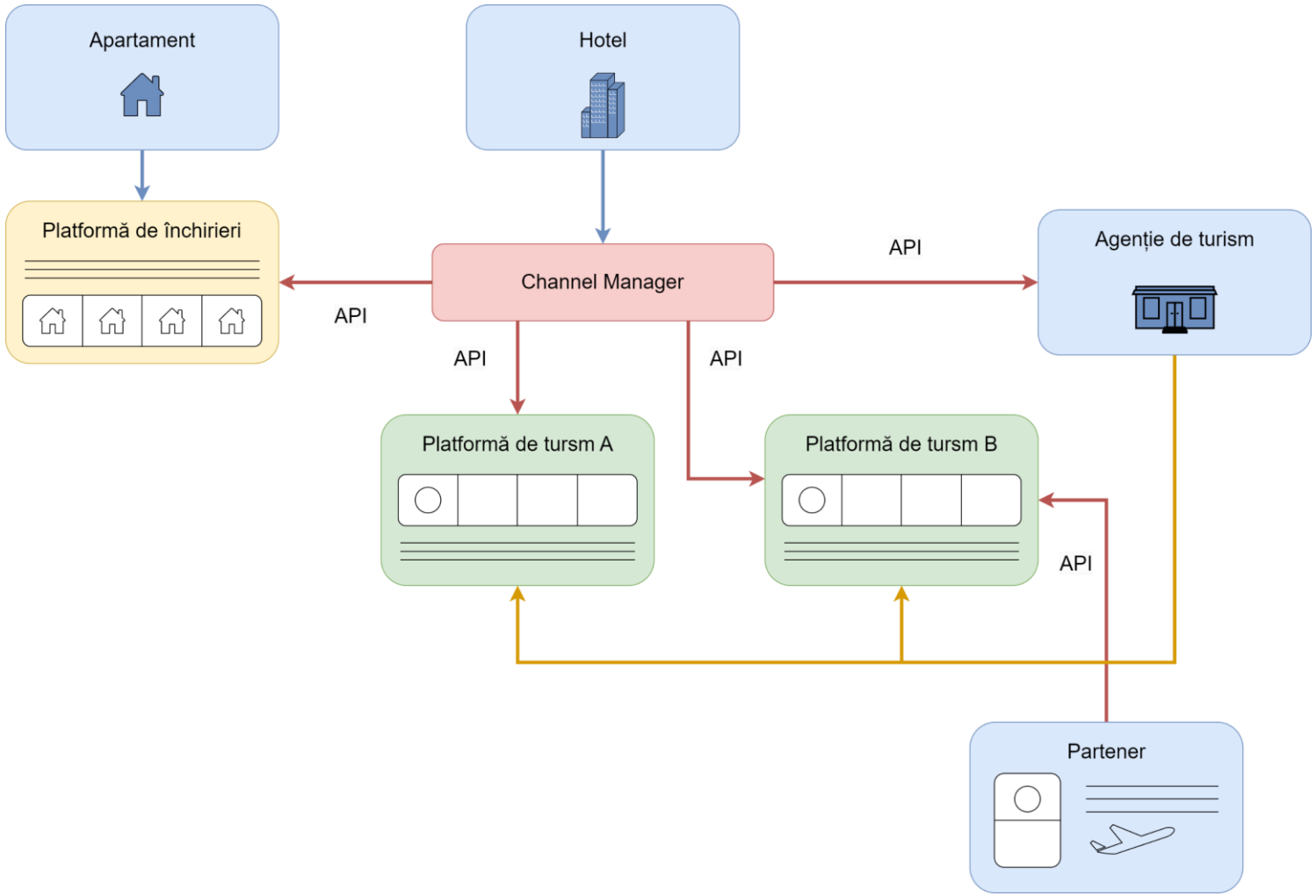
Sursa: Alex Xu, EP36: Types of Databases and Use Cases, ByteByteGo, 2023

Nivelul funcțional – API

Integrarea mai multor aplicații prin intermediul interfețelor – API

- Sincronizare date
- Interacțiuni funcționale

Studiu de caz: Ecosistemul platformelor de turism online



Nivelul interfeței utilizator

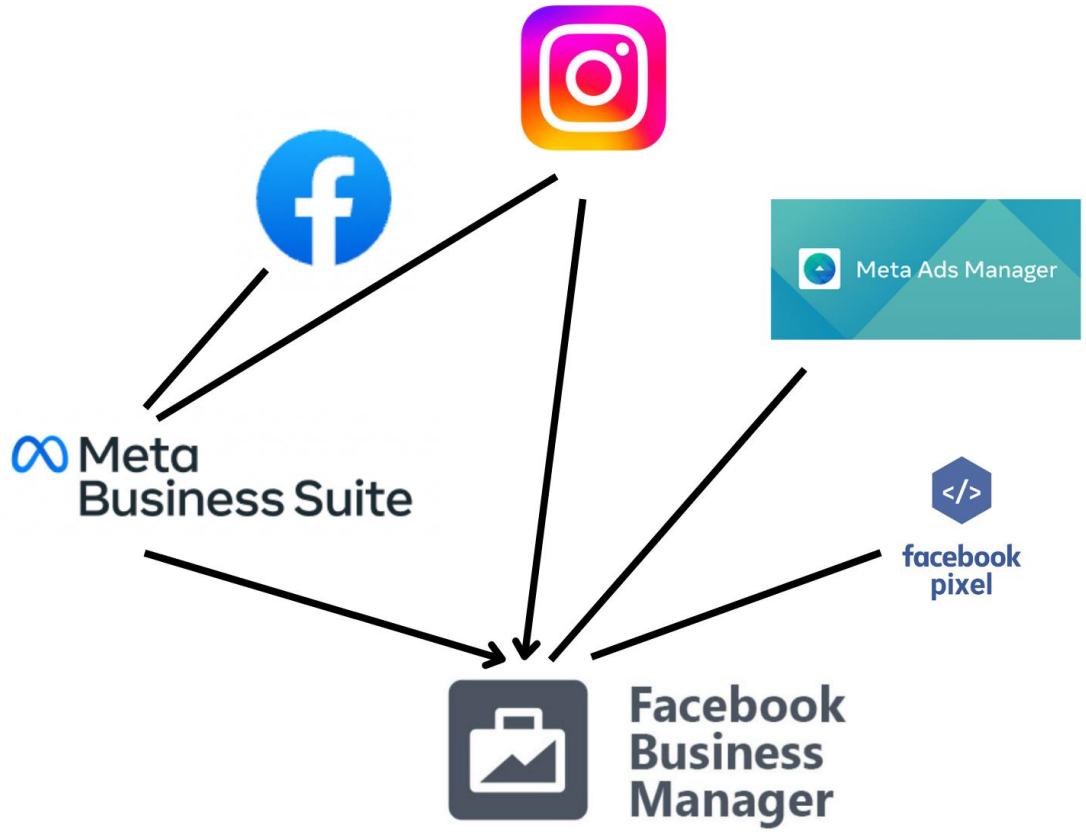
- Integrarea se face prin cumularea mai multor interfețe utilizator
- Nu necesită modificări ale aplicațiilor integrate
- Exemple
  - Widget-uri de prognoză meteo
  - Microsoft 365, Google Workspace

Studiu de caz: Meta Business Suite

Concept: Interfață centralizatoare “all-in-one” pentru gestionarea facilă a paginilor business pe Facebook și Instagram

Context: un ecosistem în continuă schimbare

Rezultat: o serie de interfețe imbricate, dificil de navigat



Sursa: Meta Business Suite vs Meta Business Manager, TwoSix Digital

Arhitecturi de aplicații software

Cum alegem arhitectura potrivită?

Analiză	Implementare
Analiza și specificarea cerințelor funcționale, non-funcționale **Identificarea constrângerilor** buget, timp, resurse, tehnologii, standarde, dependențe **Modelarea contextului** domeniu, utilizatori, stakeholderi	**Utilizarea principiilor arhitecturale** ex. SOLID, DRY, YAGNI, KISS **Utilizarea modelelor arhitecturale** ex. MVC, microservicii, event-driven, multi-nivel **Colectare de feedback** cerințe utilizatori, validare, review

Sursa: LinkedIn, 2023

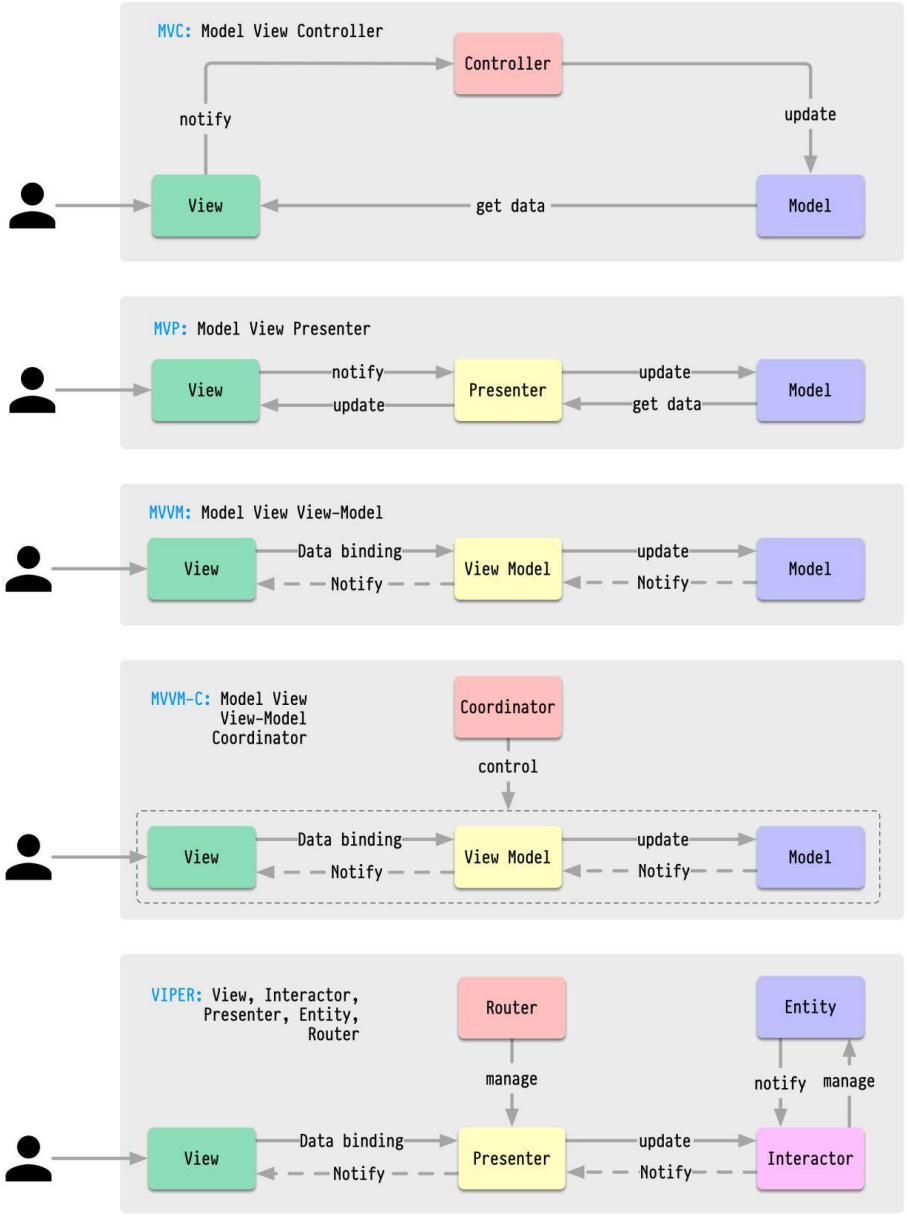
Modele arhitecturale

Modele arhitecturale

- MVC, cel mai vechi model (50+ ani)
- MVP, MVVM, MVVM-C, VIPER

Componente arhitecturale

- **Model** (M) – definește nivelul de date
- **View** (V) – responsabil pentru afișarea conținutului și interfața cu utilizatorul
- **Controller** (C) / Presenter (P) / View-Model (VM) – niveluri de legătură dintre View și Model



Sursa: ByteByteGo, 2023

Modele arhitecturale

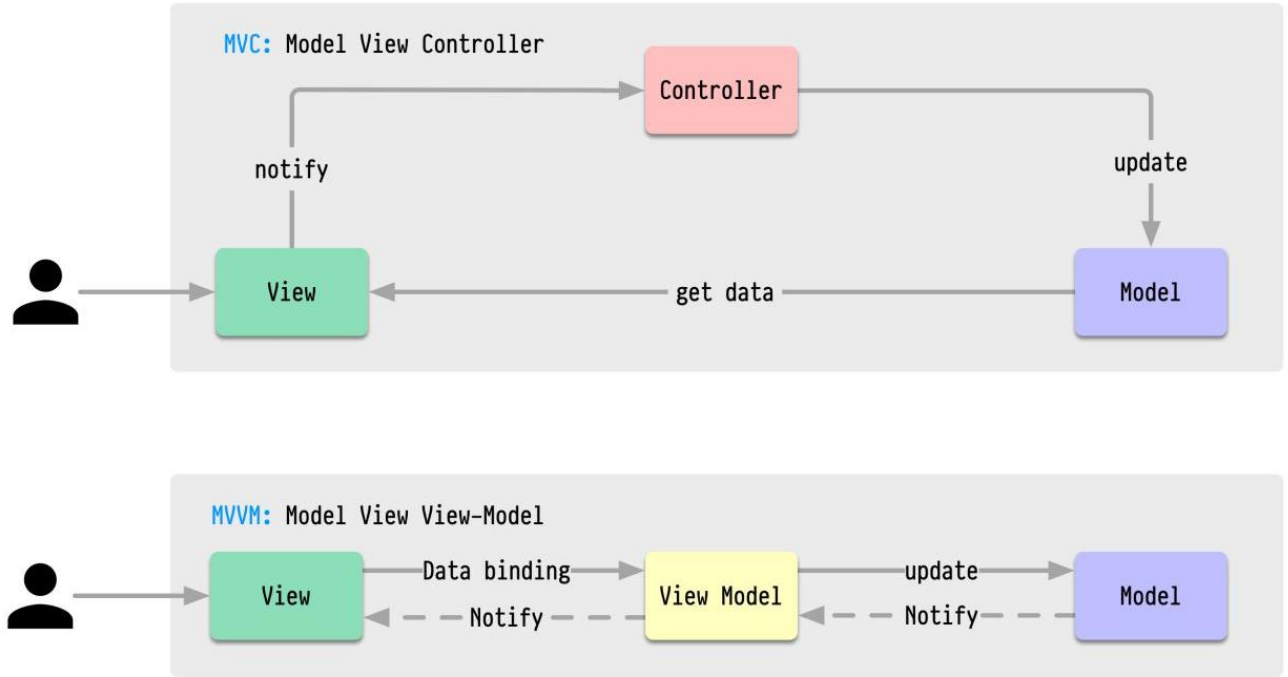
Modele arhitecturale – exemple

Generale

- MVC – ASP.NET
- MVVM – Angular

Mai specializate

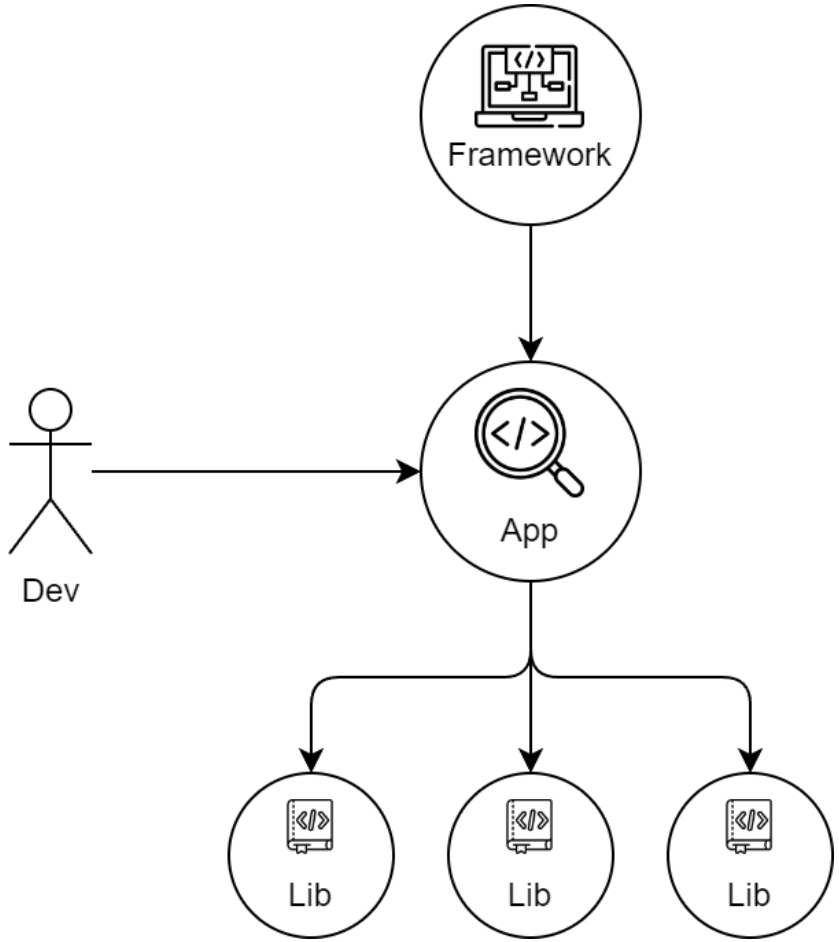
- MVVM-C – Swift
- MVP (Model-View-Presenter) – Android
- VIPER (View-Interactor-Presenter-Entity-Router) – Swift



Sursa: ByteByteGo, 2023

Framework-uri

- Un framework este mai mult decât o bibliotecă software
- Este un cadru integrat în care se dezvoltă aplicația software (structură de bază / schelet, componente reutilizabile, instrumente de proiectare / dezvoltare, etc.)

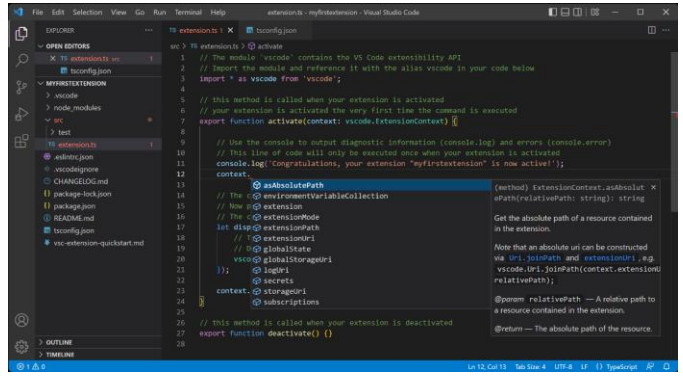


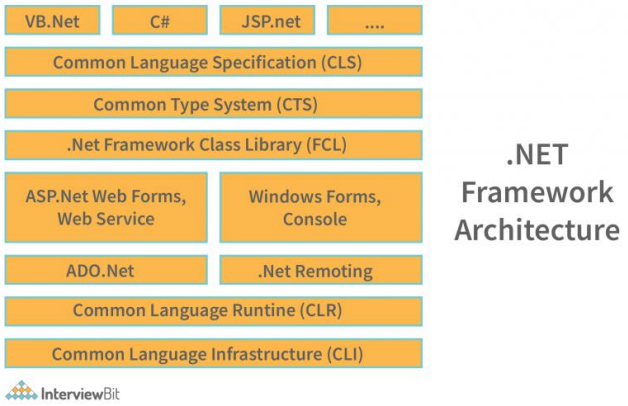
Sursa: codecademy. What is a Framework?

Framework-uri.. pe larg

Tipuri de framework-uri

- De **proiectare** software (principii, șabloane, standarde)
- De **dezvoltare** software (IDE-uri, instrumente de dezvoltare, testare, etc.)
- De **aplicație** (implementare: structură de bază / schelet, platformă: execuție / runtime)
- De **planificare / coordonare** proiect





Sursa: codecademy. What is a Framework?

Framework-uri de proiectare

Modelare (principii de modelare)

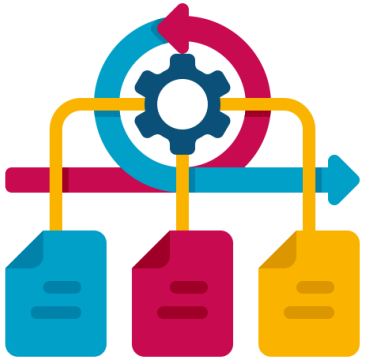
- Modele de **arhitectură**: de business, de sistem, de aplicație
- Metamodel: DSL (domain specification language)

Proiectare (design patterns)

- Modele de proiectare – **soluții concrete** pentru probleme de proiectare
- **Vezi secțiunea “design patterns”**

Implementare (standarde)

- **Directive** de proiectare elaborate de firma de proiectare
- Explicații ale principiilor și modelelor de proiectare



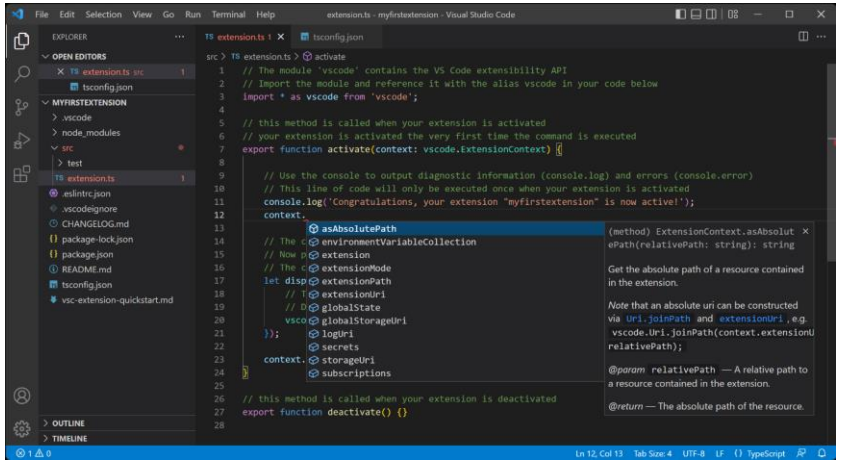
Sursa: Azure Architecture Center, Microsoft

Framework-uri de dezvoltare

Framework de dezvoltare

Componente ale mediului de dezvoltare care pun la dispoziție instrumente pentru implementare într-un mediu integrat (Integrated Development Framework)

- IDE-uri
- SDK-uri
  - Editoare de model
  - Instrumente de dezvoltare
  - Instrumente de testare, build, deployment, etc.



Sursa: Azure Architecture Center, Microsoft

Integrated Development Environment – IDE

Exemple de funcționalități integrate

Munca în echipă

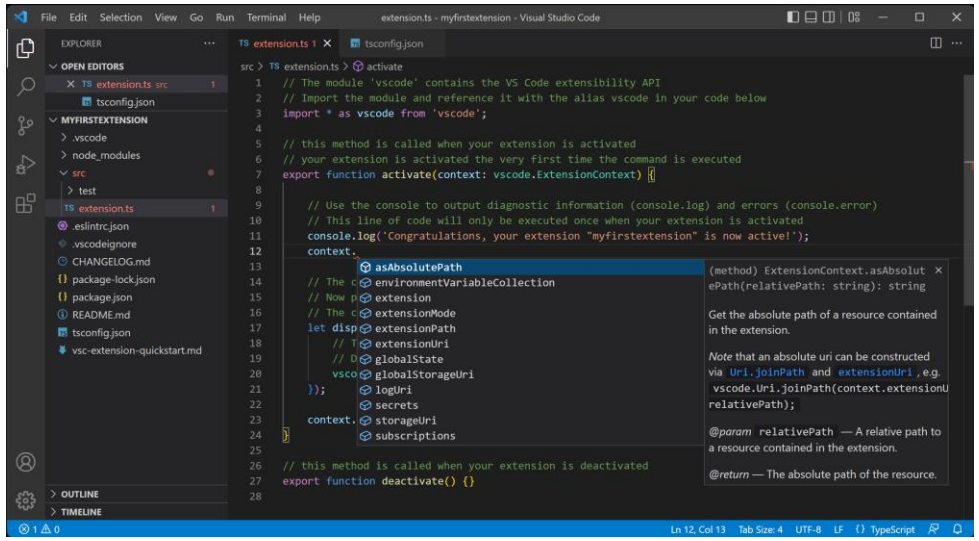
- KM/SCM – Wiki
- Bug-Tracking
- Management de proiect

Calitate

- Code coverage
- Code convention checker
- Analiza statică a codului
- Unit test

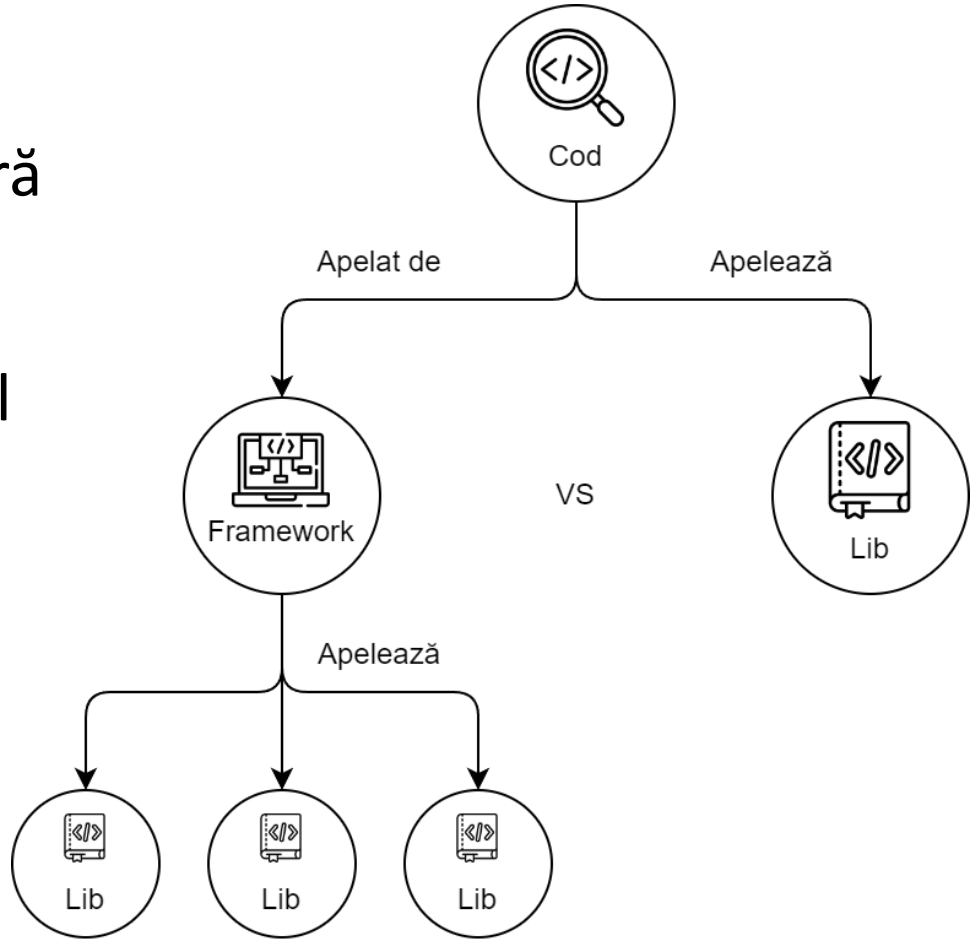
Productivitate

- Editoare /extensii specializate
- Instanțe locale (runtime container virtualization)



Framework-uri de aplicație – implementare

- Un **framework de aplicație** este o abstractizare în care un software, care oferă funcționalități generice, poate fi schimbat în mod selectiv prin adăugarea unui cod adițional scris de utilizator, generând astfel o aplicație specifică
- Framework-urile permit refolosirea de resurse și oferă un set standard de **arhitecturi de aplicații** ce sunt gata proiectate și testate



Sursa: What is a Framework in Programming? And Why You Should Use One, net solutions

Framework-uri de aplicație – implementare

Exemple de framework-uri:

Aplicații web

- Front-end: Angular, React, Vue
- Back-end: ASP.NET, Node.js, Spring Boot, Flask, Django

Website dev / Content management (CMS)

- WordPress, Drupal, Joomla, Kentico

Mobile

- React Native, Flutter, Xamarin, Ionic, Cordova

Data Science / AI

- TensorFlow, Keras, Scikit-learn, PyTorch, Apache Spark

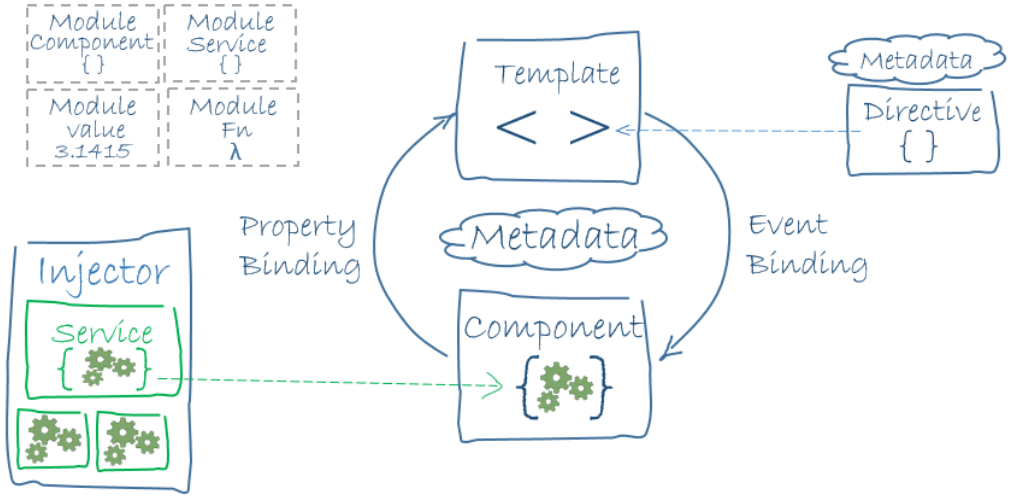
Sursa: What is a Framework in Programming? And Why You Should Use One, net solutions, 2023

Studiu de caz – Angular

De ce Angular?

- Facem la laborator 🙃
- **Arhitectură modulară** bazată pe componente
- Mecanism de rutare în aplicații de tip **Single Page Application** cu structură complexă
- **Sincronizare bidirecțională** View – Model (2-way data binding)
- **Injectarea dependențelor** (dependency injection)
- **Framework matur și stabil, comunitate mare, dezvoltat de Google**
- **TypeScript**

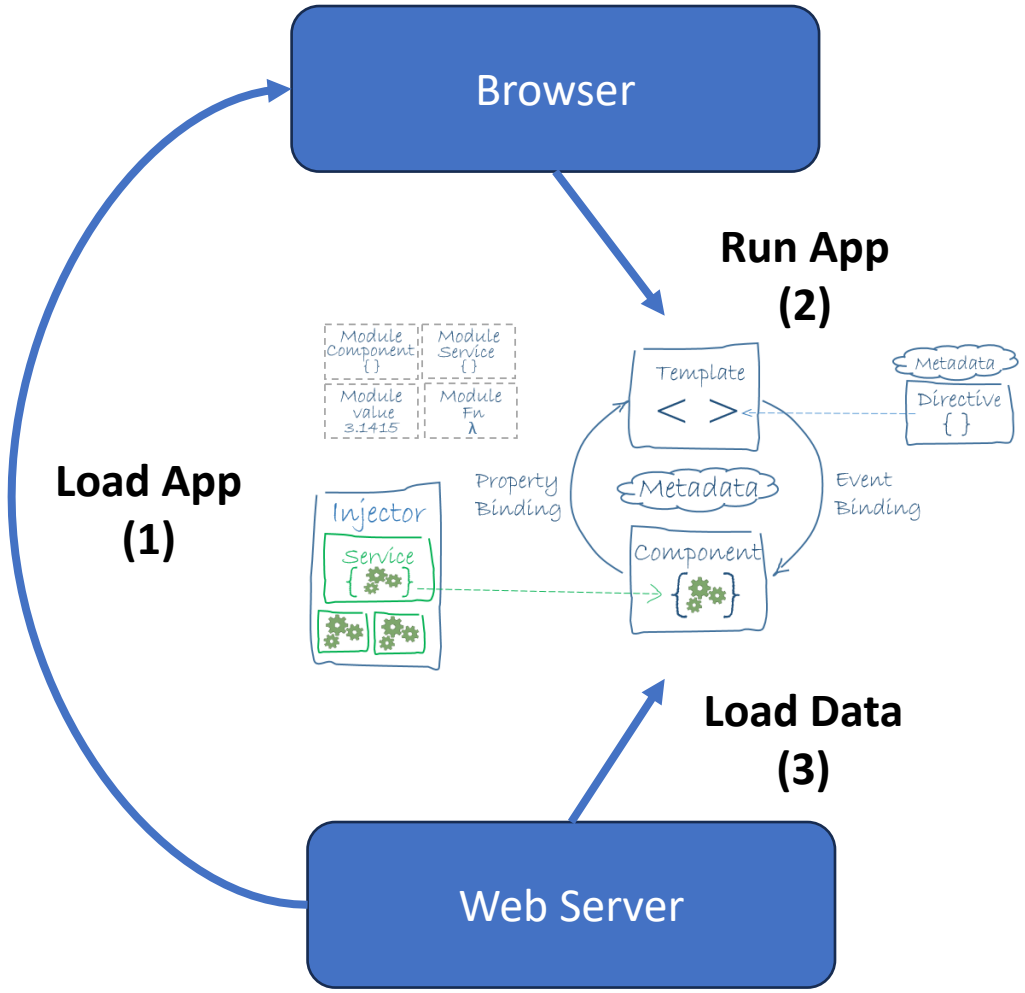
Single Page Application



Studiu de caz – Angular

Single Page Application (SPA)

- O aplicație web care rescrie în mod dinamic o pagină web curentă cu date noi de la serverul web, în loc de a încărca o nouă pagină cu totul.



Framework-uri de aplicație – platformă

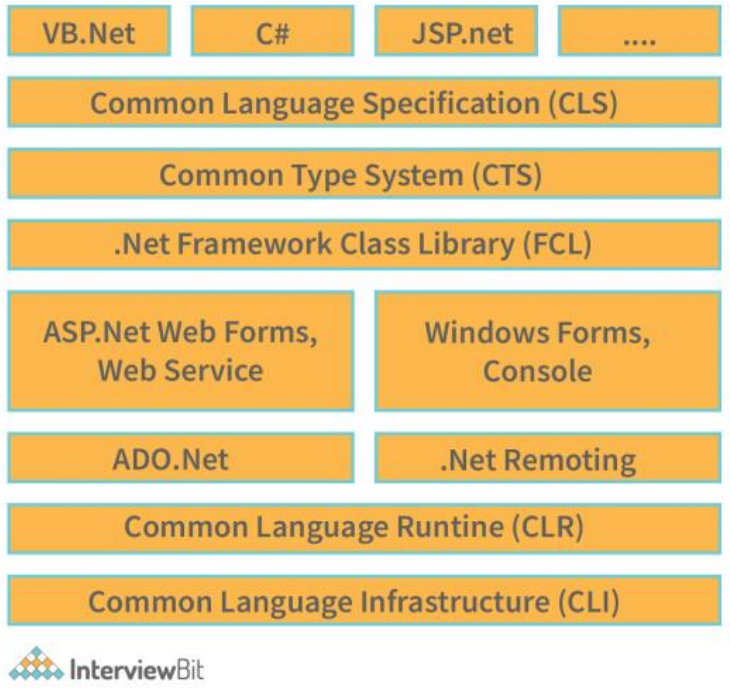
Platforme (runtime environment)

- Java Virtual Machine (JVM)
- Common Language Infrastructure (CLI)
- Common Language Runtime (.NET CLR)

Monitorizare

Tehnologii de integrare

- Enterprise Service Bus (ESB)
- Cloud Computing
- Software as a Service (SaaS)



.NET Framework Architecture

Sursa: .Net Framework Architecture – Detailed Explanation, InterviewBit

Sursa: What is a Framework in Programming? And Why You Should Use One, net solutions, 2023

Framework-uri de planificare/coordonare

Planificare activități

- Diagrame GANTT
- Work Breakdown Structure (WBS)

Metode de dezvoltare

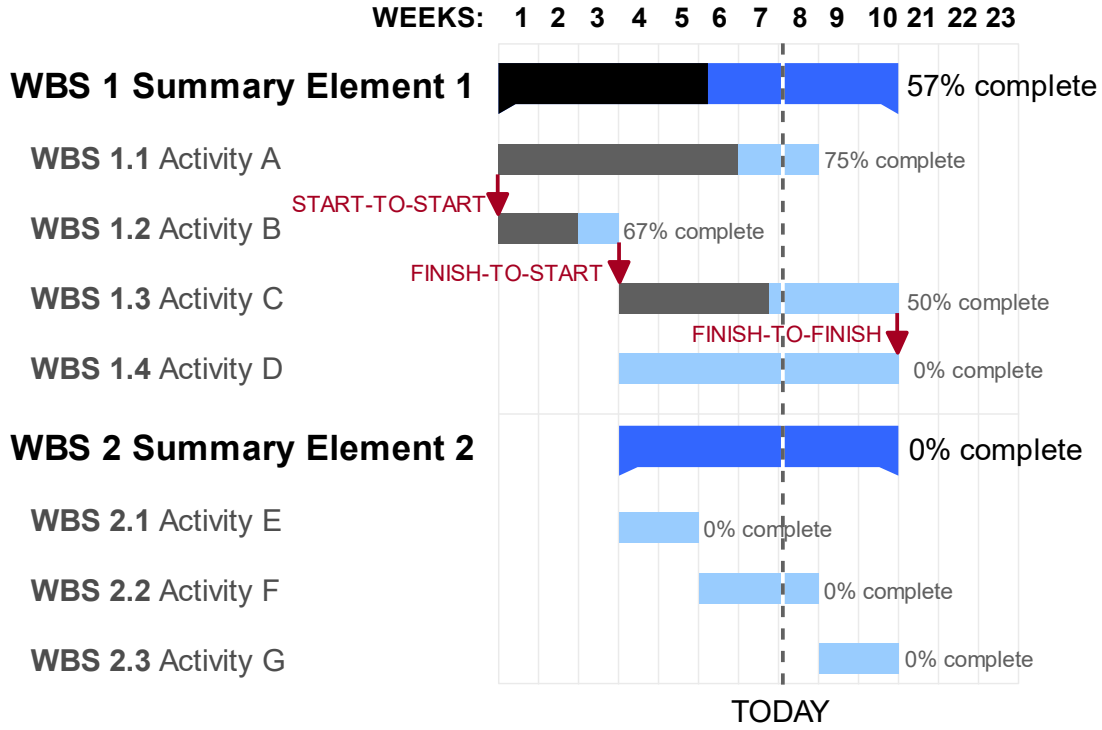
- Waterfall
- Agile

Cicluri de dezvoltare/actualizare

Documentație

Managementul ciclului de viață

- Sisteme de versionare

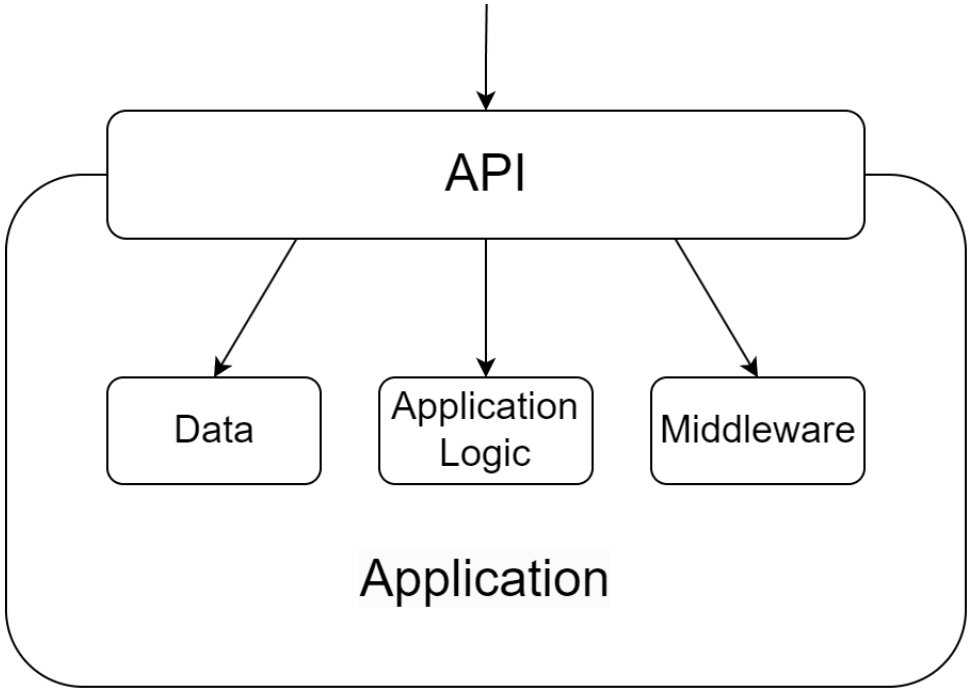


Sursa: What is a Framework in Programming? And Why You Should Use One, net solutions, 2023

Integrarea aplicațiilor prin interfețe – API

Application Programming Interface

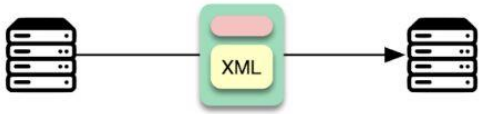
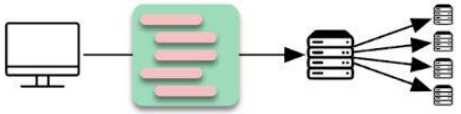
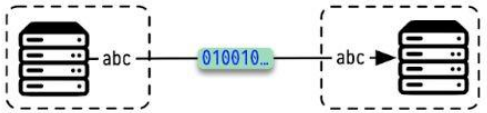
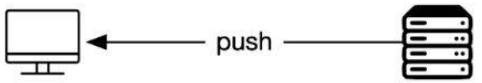
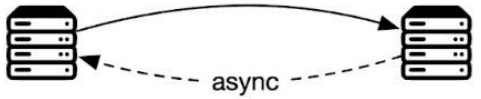
- Un API este un mecanism bine definit pentru conectarea la una sau mai multe resurse cum ar fi serverul de aplicație, nivelul middleware sau baza de date
- API-ul permite dezvoltatorului să invoce serviciile acestor entități pentru a obține un rezultat
- Reprezintă principala metodă de integrare la nivel de logică de aplicație / procese de afaceri



Arhitectura unui API

- API-urile se folosesc și pentru **interfațarea** dintre mai multe aplicații
- Un **endpoint** reprezintă o **resursă adresabilă** în cadrul unui API (de ex. printr-o cerere HTTP)
- Pentru o interfațare (eficientă)
  - Este esențial să avem un **model arhitectural compatibil**
  - și potrivit **cazurilor de utilizare**

API Architecture Styles — ByteByteGo.com

Style	Illustration	Use Cases
SOAP		XML-based for enterprise applications
RESTful		Resource-based for web servers
GraphQL		Query language reduce network load
gRPC		High performance for microservices
WebSocket		Bi-directional for low-latency data exchange
Webhook		Asynchronous for event-driven application

Sursa: ByteByteGo, 2023

Arhitectura unui API – API First

În 2002, Jeff Bezos a definit o primă formă a conceptului API First – sistemele devin un fel de piese de Lego **reconfigurabile**

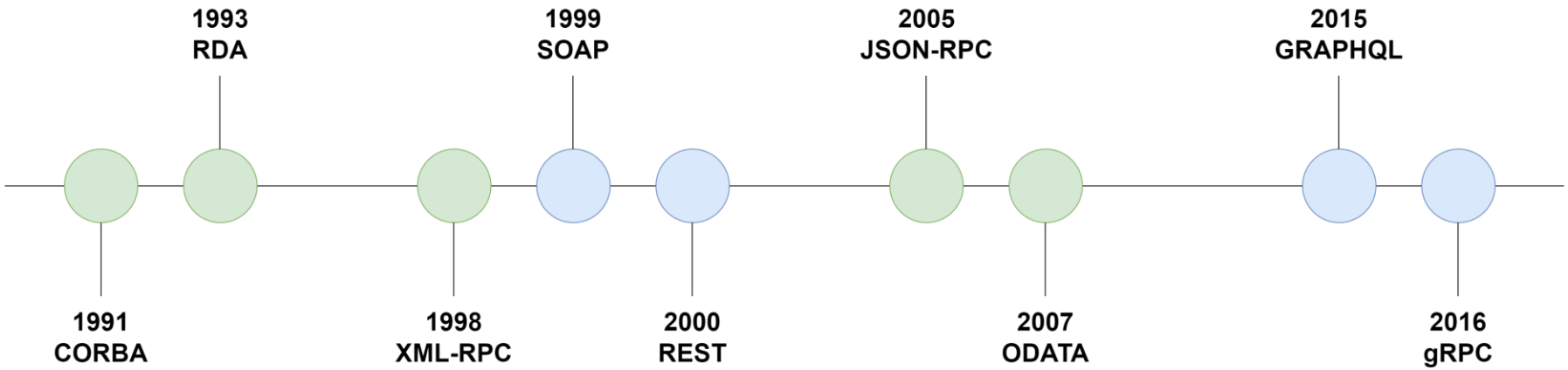
- Datele și funcționalitățile trebuie să fie **expuse** prin API-uri
- Comunicarea dintre echipe trebuie să fie realizată prin API-uri
- Nu pot exista alte canale/scurtături
- Alegerea tehnologiilor este secundară
- API-urile trebuie să fie **externalizabile**

Sursa: Alex Xu, Mastering the Art of API Design, ByteByteGo, 2023

Arhitectura unui API – Modele arhitecturale

Modele arhitecturale de API

- **SOAP** – XML, aplicații enterprise
- **RESTful** – HTTP, servicii web
- **GraphQL** – cereri structurate
- **gRPC** – comunicație microservicii
- **WebSocket** – aplicații de timp real
- **Webhook** – HTTP, notificări /callback



Sursa: Alex Xu, SOAP vs REST vs GraphQL vs RPC, ByteByteGo, 2022

Arhitectura unui API – Modele arhitecturale

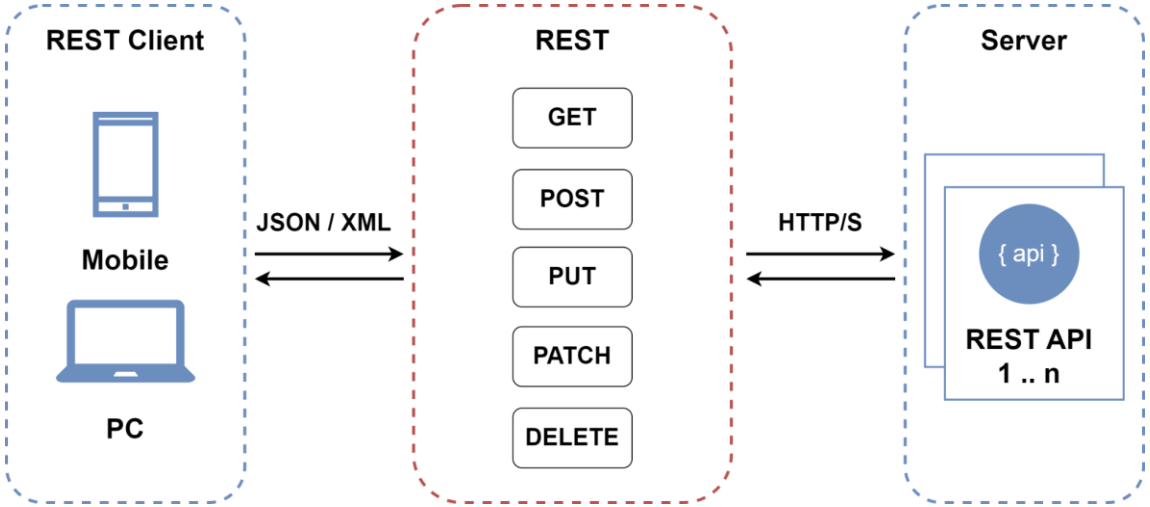
	SOAP	**REST**	**GraphQL**	**RPC**
caracteristică	Mesaje încapsulate	6 principii arhitecturale	Schemă și tipuri de date	Apel proceduri locale
format	XML	XML, JSON, HTML, text	JSON	JSON, XML, Protobuf, Thrift, FlatBuffers
dificultate de învățare	dificil	ușor	mediu	ușor
comunitate	mică	mare	În creștere	mare
cazuri de utilizare	- Procesatoare de plăți - Autentificare - Sisteme CRM - Sisteme legacy	- API-uri publice - Aplicații simple	- API-uri mobile - Sisteme complexe - Microservicii	- API-uri de comandă - Comunicație microservicii în sisteme complexe

Sursa: Alex Xu, SOAP vs REST vs GraphQL vs RPC, ByteByteGo, 2022

Arhitectura unui API – HTTP

REpresentational State Transfer

- Principii arhitecturale
  - Interfață uniformă (independentă de dispozitiv sau aplicație)
  - Nu păstrează starea (stateless)
  - Caching (stocare temporară)
  - Arhitectură client-server
  - Arhitectură pe mai multe niveluri
- Cazuri de utilizare
  - API-uri publice/de uz general

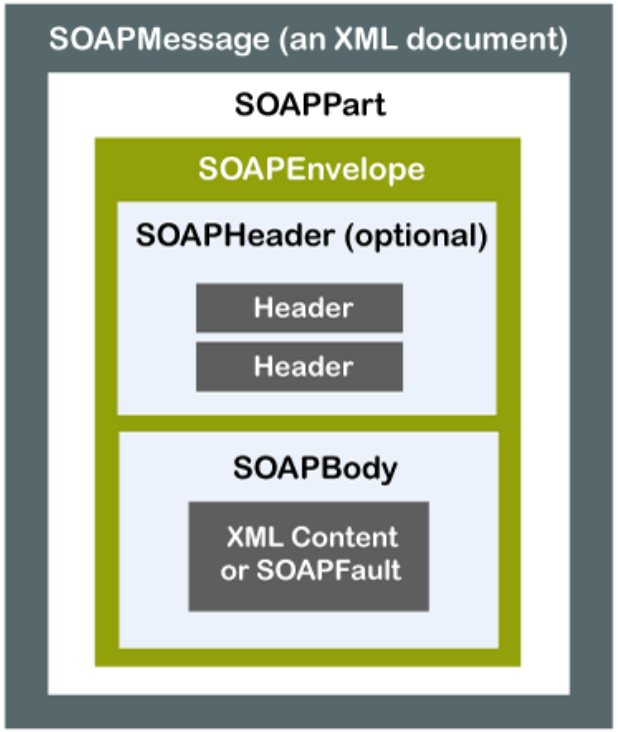


Sursa: Comparing API Architectural Styles: SOAP vs REST vs GraphQL vs RPC, Altexsoft, 2020

Arhitectura unui API – SOAP

Simple Objects Access Protocol

- Mesaje în format XML prin HTTP
  - Împachetare (envelope)
  - Antet (header)
  - Conținut (body)
  - Indicatori de eroare (fault)
- Cazuri de utilizare
  - Transmisie ultra securizată în cadrul întreprinderilor/transferuri de date securizate



Sursa: Introduction of SOAP and REST Web Services, Javatpoint

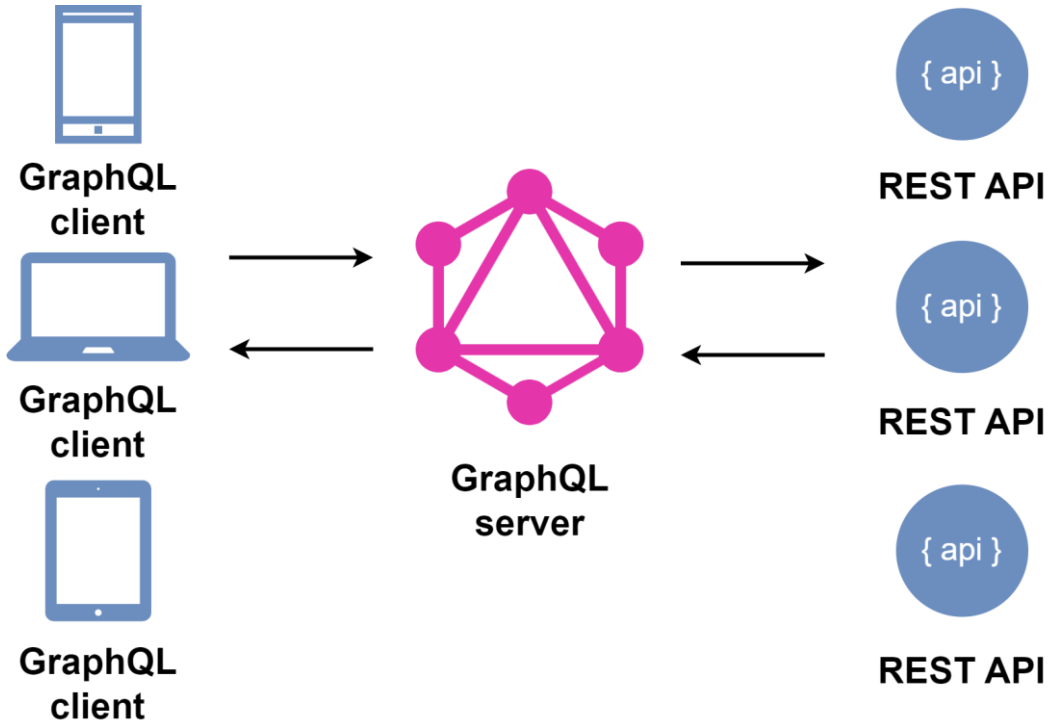
Sursa: Comparing API Architectural Styles: SOAP vs REST vs GraphQL vs RPC, Altexsoft, 2020

Arhitectura unui API – GraphQL

- Cereri structurate prin sintaxă
  - Schema resurselor disponibile
  - Interpretare cereri structurate
  - Returnare resurse cerute (ex. date)

Cazuri de utilizare

- Cereri complexe în sisteme pe bază de microservicii – agregarea datelor din mai multe surse într-un mod structurat



Sursa: Comparing API Architectural Styles: SOAP vs REST vs GraphQL vs RPC, Altexsoft, 2020

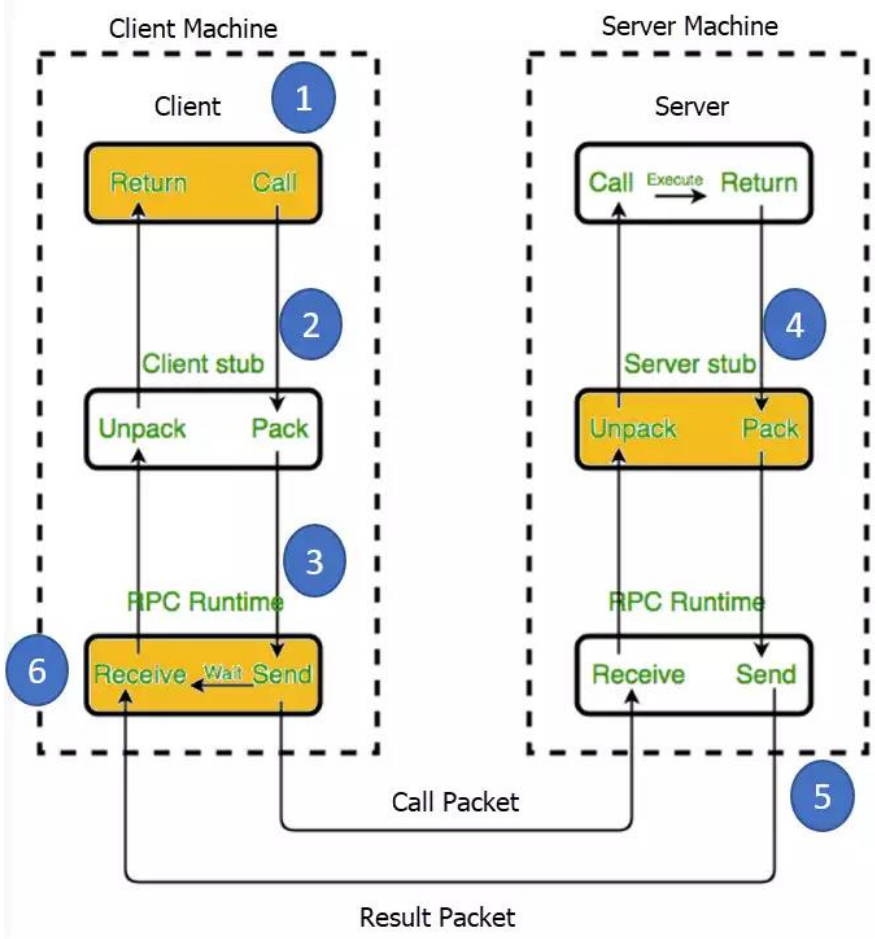
Arhitectura unui API – RPC

RPC – Remote Procedure Call

- inițial XML-RPC
- actual JSON-RPC, sub forma unui API HTTP
- gRPC – dezvoltat de Google în 2015

Cazuri de utilizare

- Transmitere de comenzi către un sistem remote
  - ex. Slack – join channel, leave channel, send message



Sursa: Comparing API Architectural Styles: SOAP vs REST vs GraphQL vs RPC, Altexsoft, 2020

Arhitectura unui API – Webhook

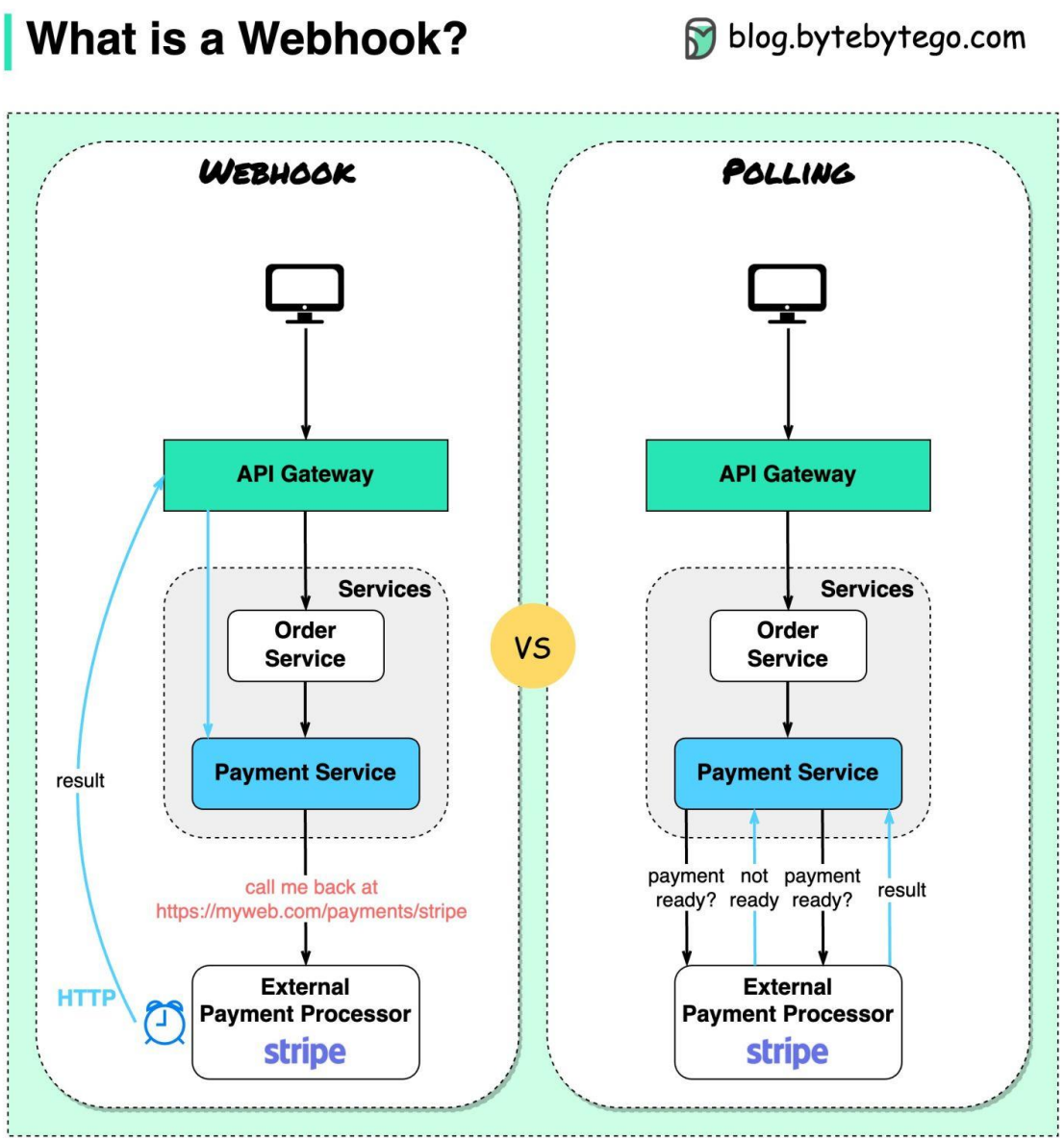
Studiu de caz: procesator de plăți

- Confirmarea plăților nu se face instant (ex. autorizări suplimentare, 2FA)

Modele de comunicație

- **Polling –** interogare periodică a serviciului de plăți pentru confirmare
- **Webhook –** serviciul de plăți apelează înapoi aplicația atunci când este confirmată plata

Care dintre acestea este mai eficient în condiții de utilizare intensă?



Sursa: ByteByteGo, EP65: What Is A Webhook, 2023

Arhitectura unui API – REST vs GraphQL

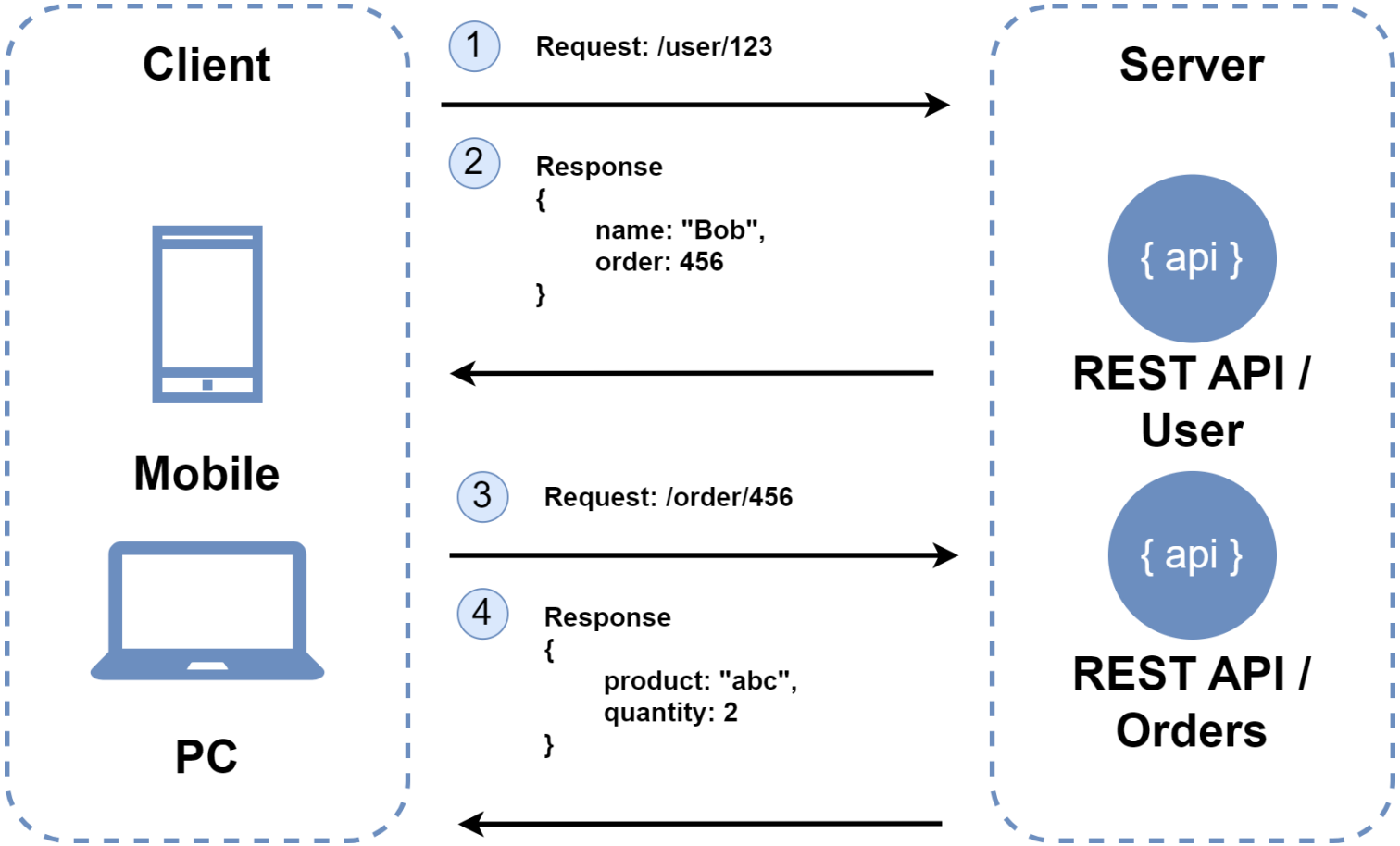
Comparație

- GraphQL – dezvoltat de Meta, limbaj de specificare / agregare a resurselor solicitate de la unul sau mai multe API-uri
- În cazul unui REST API, aceste resurse ar trebui agregate fie
  - la nivel de client (mai multe cereri)
  - la nivel de API Gateway (mai multe endpoint-uri)

Image by Freepik

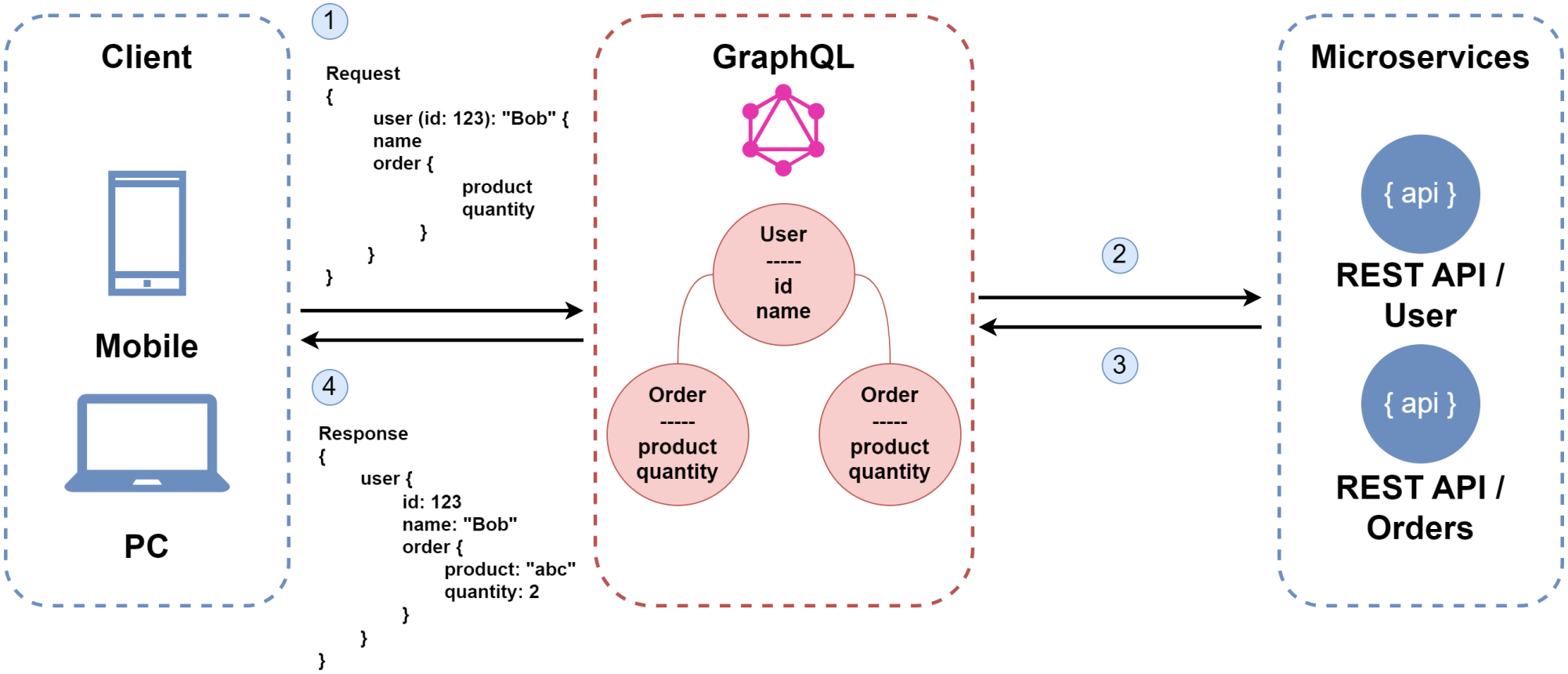
Sursa: Alex Xu, What is GraphQL? Is it a replacement for the REST API?, LinkedIn, 2023

Arhitectura unui API – REST vs GraphQL



Sursa: Alex Xu, What is GraphQL? Is it a replacement for the REST API?, LinkedIn, 2023

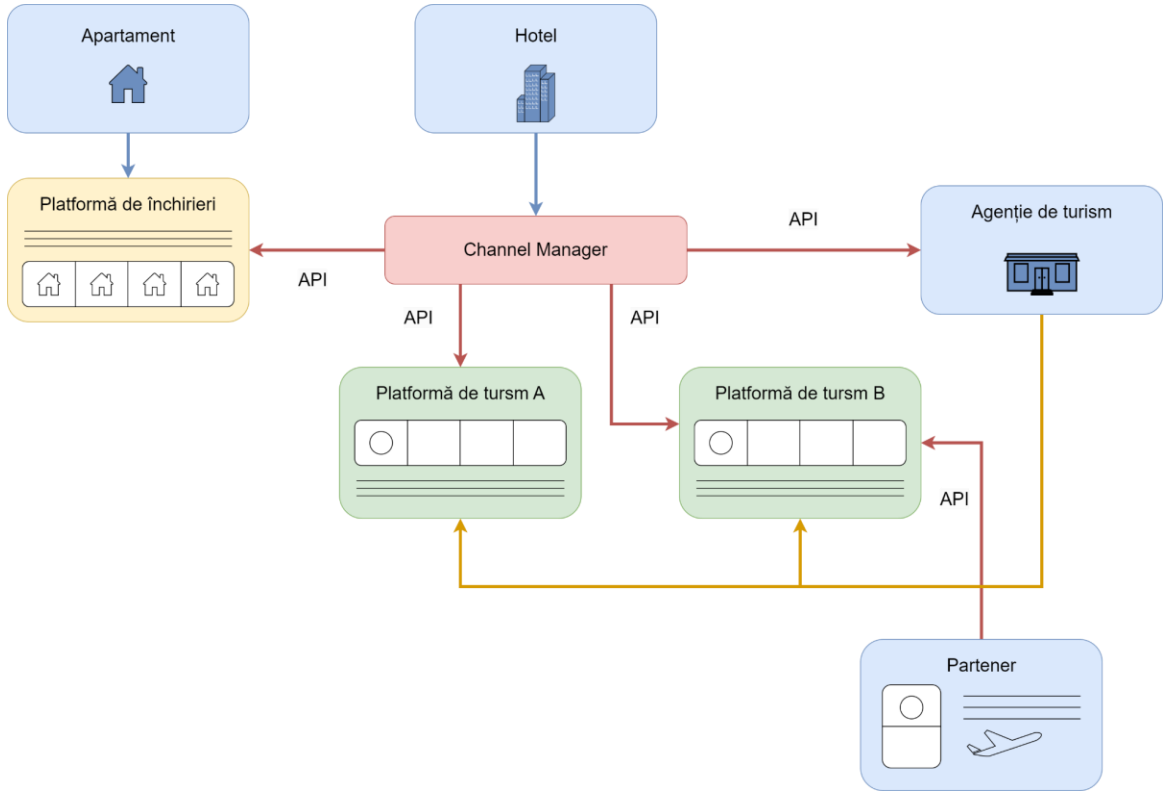
Arhitectura unui API – REST vs GraphQL



Sursa: Alex Xu, What is GraphQL? Is it a replacement for the REST API?, LinkedIn, 2023

Studiu de caz – Integrări complexe

- Se dorește integrarea **sistemului de rezervări al unui hotel** cu ecosistemul de turism online (platforme online, agenții de turism, parteneri)
- Channel Manager – componentă **specifică** domeniului rezervărilor online, care integrează multiplele canale de distribuție pentru gestionarea rezervărilor provenite din mai multe surse diferite

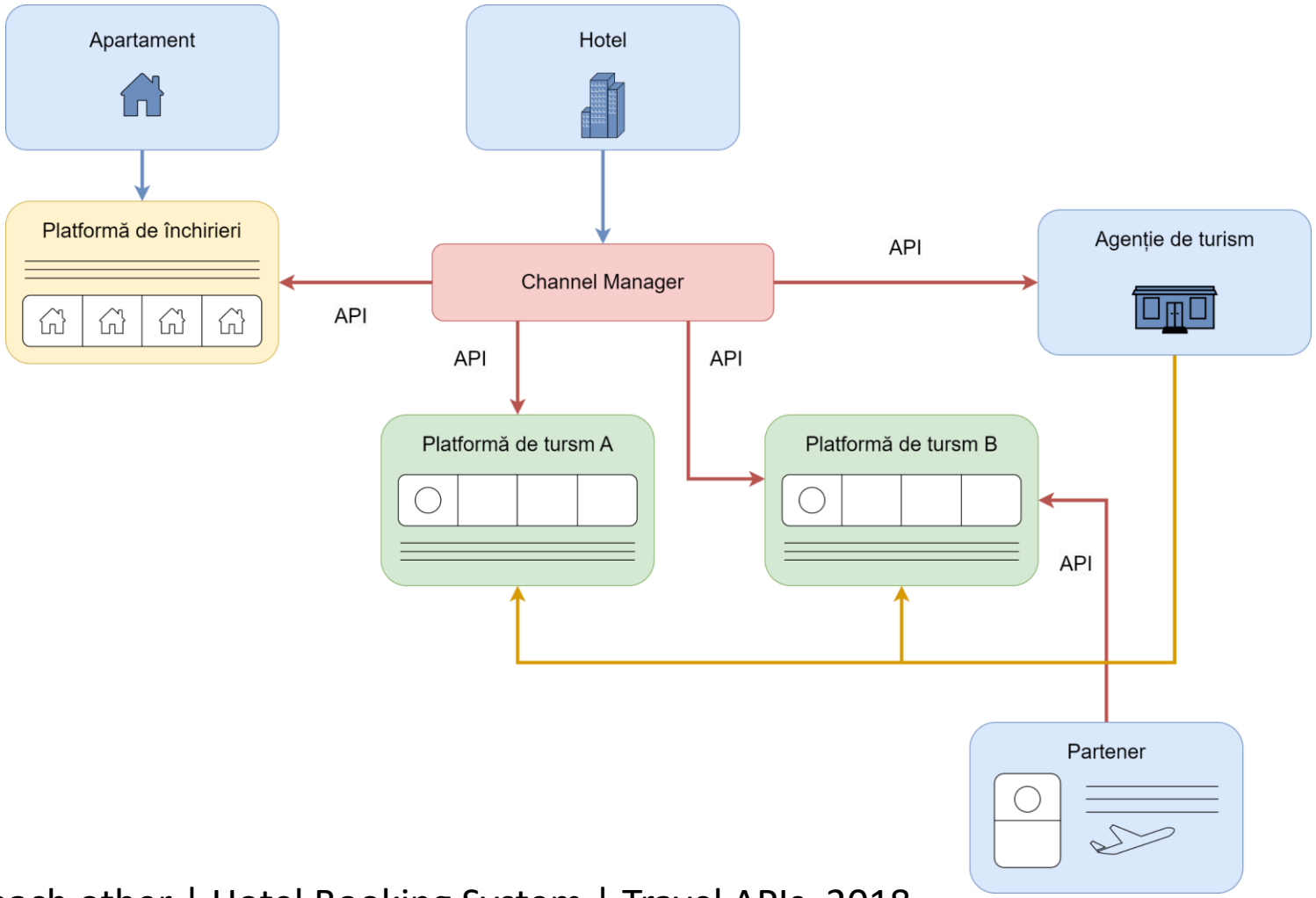


Sursa: How travel systems talk to each other | Hotel Booking System | Travel APIs, 2018

Studiu de caz – Integrări complexe

Problemă

- Identificați tipurile de integrări din schemă și descrieți funcționarea sistemului integrat
- Propuneți soluții (alternative) pentru o **integrare mai eficientă**



Sursa: How travel systems talk to each other | Hotel Booking System | Travel APIs, 2018

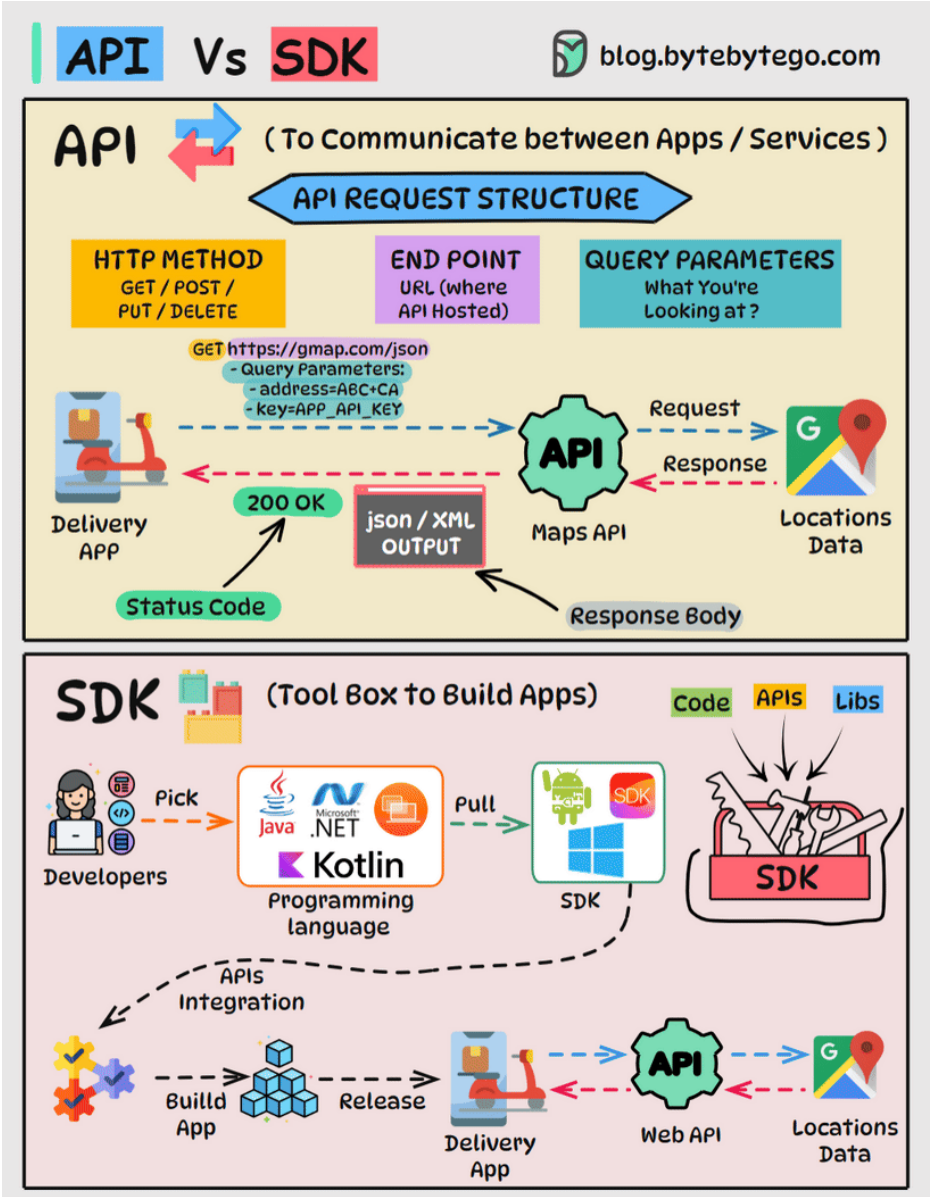
API vs SDK

API

- Interfețe și protocoale de comunicație între aplicații software diferite

SDK

- Colecție de instrumente, biblioteci, secvențe de cod și documentație pentru realizarea aplicațiilor pentru o anumită platformă, framework, sau hardware



Sursa: ByteByteGo, EP83: Explaining 9 Types of API Testing, 2023

API vs SDK

	API	**SDK**	**Bibliotecă**	**Framework**
Scop	Descrie interacțiunile dintre componente /aplicații diferite	Abstractizează implementările specifice	Încapsulează funcții și secvențe de cod reutilizabile	Modelează o structură complexă de biblioteci /servicii software
Utilizare	Aplicațiile interacționează prin API-uri	Aplicațiile includ SDK-uri	Aplicațiile includ biblioteci	Aplicațiile sunt construite pe baza unui Framework
Structură	Colecție de endpoint-uri de tip request – response	Colecție de instrumente și biblioteci specifice platformei	Colecție de funcții, clase, interfețe	Colecție de biblioteci și modele arhitecturale structurate
Exemple	• Google Maps API • ChatGPT API • Twitter API	• Android SDK • Java Dev. Kit	• Chart.js • Socket.io • JQuery	• .NET Framework • Spring Boot • Angular

Întrebări?